产业互联网时代的单笔高可用与鲁棒性

已晨，2023/05

个人简介

郭凤钊（花名：已晨）

- 专注于高可用及架构领域12年
- 菜鸟「高可信架构」的设计与实践，让用户更加信赖菜鸟
- 连续多年担任菜鸟双11、618大促技术队长
- 曾任CAINIAO SRE技术平台TL；用技术产品，让SRE的工作更简单

分享主题：《产业互联网时代的单笔高可用与鲁棒性》

目录

什么是『产业互联网』？

以数据为关键要素，以数字技术与实体经济深度融合为主线，加强数字基础设施建设，完善数字经济治理体系，协同推进数字产业化和产业数字化，赋能传统产业转型升级，培育新产业新业态新模式，不断做强做优做大我国数字经济。--《“十四五”数字经济发展规划》，2022年1月

数字产业化

为产业数字化提供数字技术、产品、服务、基础设施和解决方案，以及**完全依赖于数字技术、数据要素**的各类经济活动。

云厂商、电子商务、网约车、短视频、社交媒体等

消费互联网

产业数字化

利用数据与数字技术对**传统产业**进行升级、转型和再造的过程。（用数智化/互联网的技术和手段，让传统产业**降本增效**）

智慧农业、智能制造、智能交通、智慧物流等

产业互联网

菜鸟，客户价值驱动的全球化产业互联网公司

菜鸟“5美金10日达”：中国卖家花5美金送到全球20个国家，10个工作日到货（用互联网技术和手段，做好产业，降本提效）

UPS、FedEx、DHL：50美元，3-5日达

邮政公司：3-5美元，无法准确估计到货时间

目录

什么是高可用？

系统可用时间与“本来”可用总时间的比率，通常用“几个9”描述，代表**年度不可用时长**

系统可用性%	不可用时间/年	不可用时间/月
90%	36.5 天	72 小时
99%	3.65 天	7.20 小时
99.9%	8.76 小时	43.8 分
99.99%	52.56 分	4.38 分
99.999%	5.26 分	25.9 秒

适用于「消费互联网」，关注消费者的感官体验。业务生命周期围绕“系统服务”展开，要求「系统」高可用

如何做系统高可用？

- MTBF: Mean Time Between Failure，平均故障间隔
- MTTR: Mean Time To Repair，平均恢复时长

增加MTBF

通过增加冗余，提升整体系统可靠性&容错性

- 防止单点过载
- 防止级联故障
- 慎对重试&缓存

降低MTTR

从事故管理、变更风控、监控运维等方面，增加系统可维护性

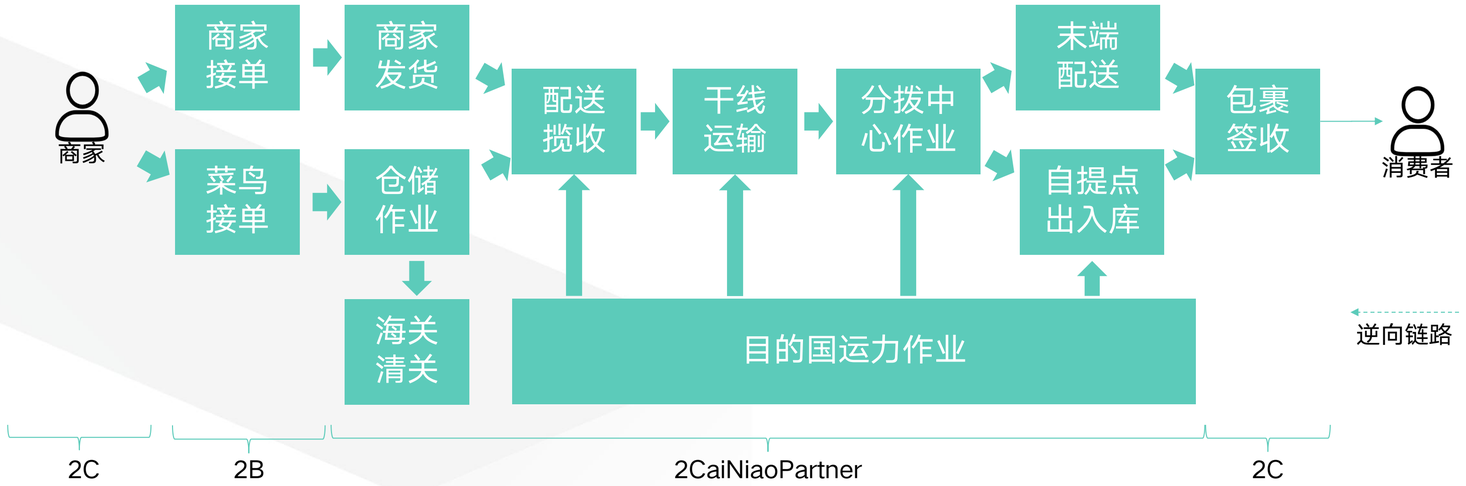
- 故障排查&应急响应
- 变更&发布管理
- 日志&监控告警

// 阿里巴巴安全生产提出的“1-5-10”是降低MTTR的典型实践

产业互联网对高可用的要求更高

菜鸟典型业务场景：包裹履约

产业互联网围绕生产过程，业务生命周期以『单』作为核心实体展开



- **任何一个**包裹异常都可能客诉/资损
- 经过的节点越多，整体失败的概率越大
- 菜鸟系统不可用类故障占比仅50%+；工单中的必填单号字段；从现实中呼应对单笔高可用判断

从「系统」高可用到「单笔订单」或「单笔请求」的高可用

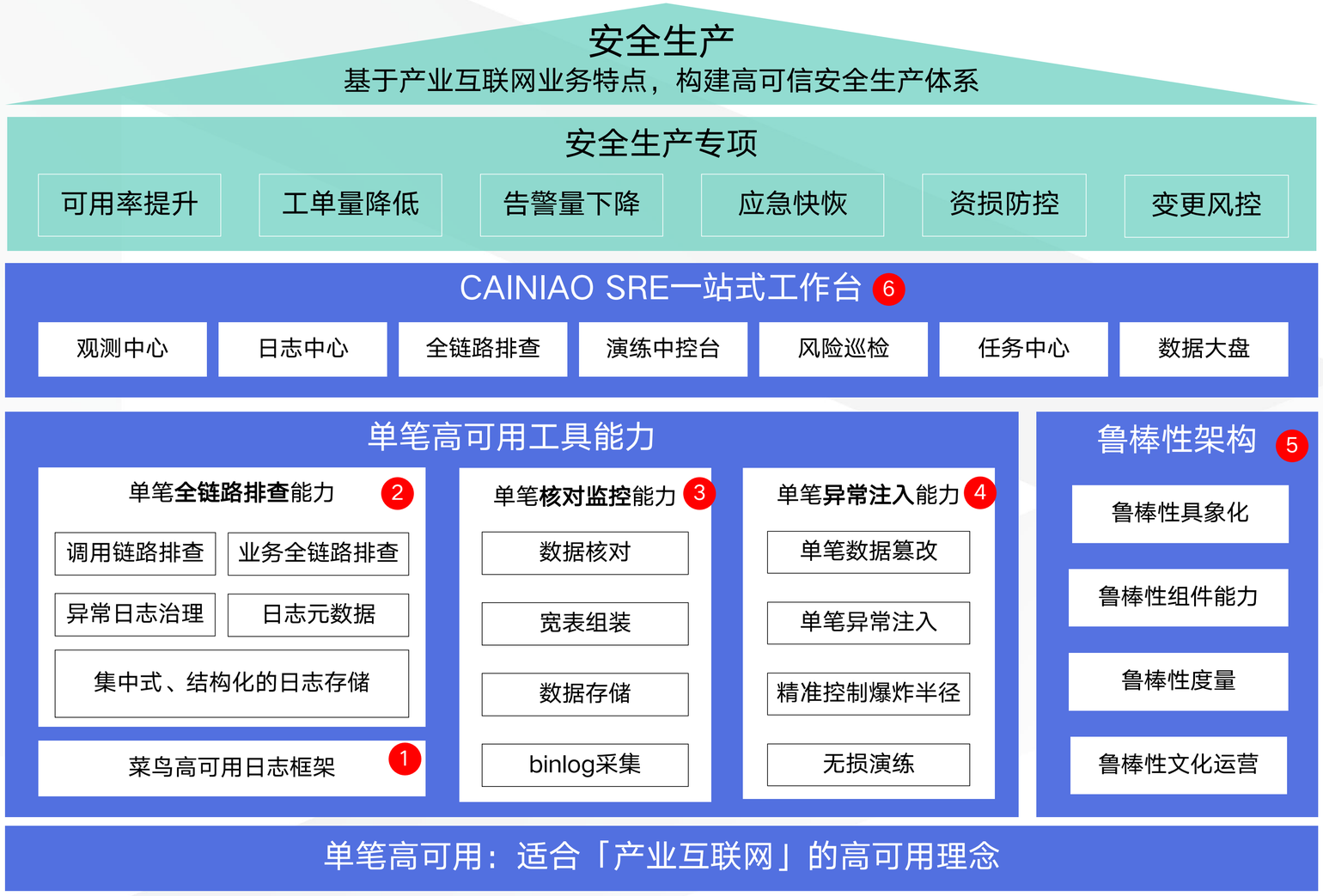
产业互联网对高可用的要求更高，挑战更大

消费互联网与产业互联网高可用策略对比

高可用策略	消费互联网的高可用策略	产业互联网的高可用策略	「单笔高可用」技术能力支撑
日志 Logging	事件 用户身份（cookie、sessionId、userId） 系统表现、错误码	事件、用户身份、系统表现、错误码 业务单号 单据状态	高可用日志框架
链路追踪 Tracing	调用链路（一次系统RPC调用，traceId唯一）	调用链路（一次完整RPC调用，traceId唯一） 业务全链路（多次RPC调用，业务单号唯一）	单笔异常的全链路排查能力
监控 Metrics	指标监控（流量、错误、延时、饱和度等）	指标监控（流量、错误、延时、饱和度等） 卡单监控 上下游数据一致性监控	单笔核对类监控能力
演练/故障注入	系统服务 容器、宿主机 网络、机房等基础设施	系统服务 容器、宿主机、网络、机房等基础设施 卡单异常注入 单据数据篡改	单笔异常的无损注入能力
应急防御	限流（对上游） 降级（弱依赖、非重要功能） 重启 切流（按地域、运营商） 回滚 扩容	限流（对上游） 降级（非重要功能） 重启 切流（按照业务ID） 回滚、扩容 控速、hold单	鲁棒性架构
变更灰度	蓝绿发布、金丝雀发布、灰度发布	蓝绿发布、金丝雀发布、灰度发布	-
压测	构造高并发	构造高并发（蓄洪）	-

1. 产业互联网的高可用策略，在消费互联网的基础上，有自己的特点（绿色文字标注）
2. 产业互联网高可用的关键能力，会围绕着可观测（监控、链路、日志）、鲁棒性架构、故障演练展开

菜鸟单笔高可用体系建设大图



理念：面向失败设计，由下到上，由外到内

1、构建统一的、结构化日志组件，为「可观测」打下基础

2、以工单作为突破口，建设单笔全链路排查能力（**被动排查**）

3、**主动识别**单笔异常，在用户感知前解决

4、建设“真实”、“无损”的单笔异常注入能力，对高可用做反向验证

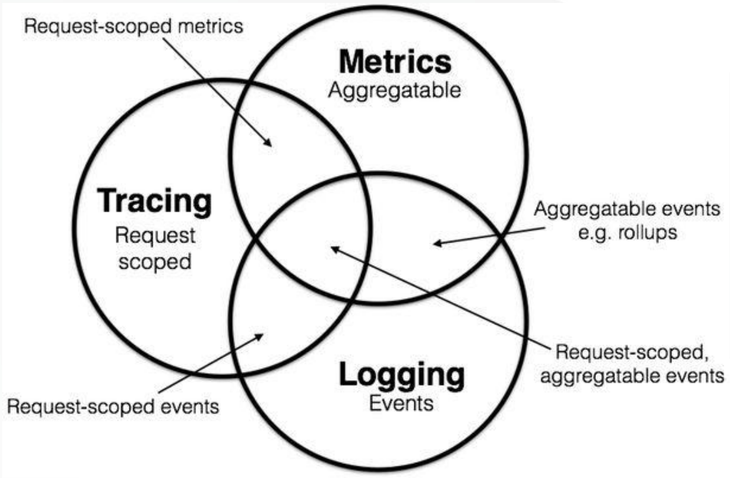
5、提升系统架构的鲁棒性，对故障**主动防御**，提升用户体验

6、以「好用的」产品透出技术能力

目录

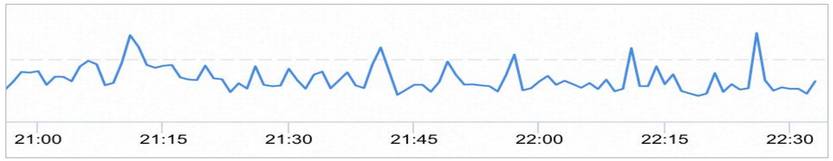
1）菜鸟『高可用日志框架』，可观测基础

可观测性三大支柱：



在企业落地挑战：割裂、各自为战、监控配置随意、日志输出随意

Metrics
监控团队



Tracing
中间件团队

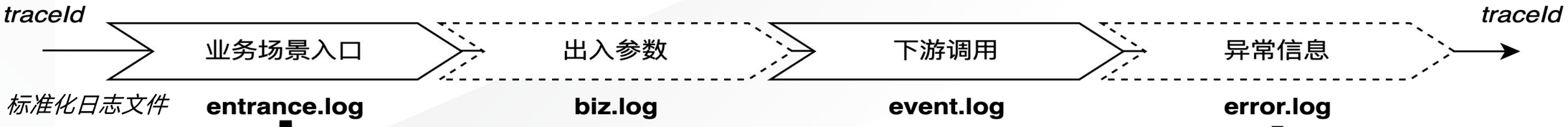
Logging
研发团队

- 监控
- 链路
- 存储

结构化日志

1. *结构统一*
2. *易于切分*

高可用日志框架 = 打印结构化日志 + 打通可观测体系



标准化日志文件

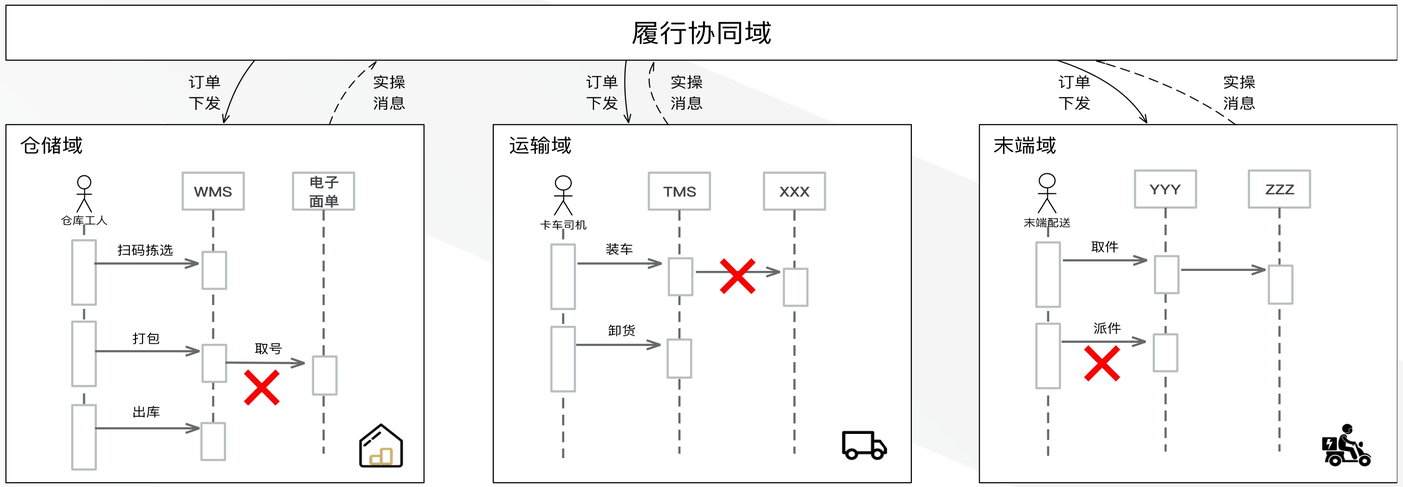
entrance.log | **biz.log** | **event.log** | **error.log**

时间戳	业务场景	是否成功	耗时	错误码	错误信息	请求id	调用序号
timestamp	eventCode	success	rt	errorCode	errorMsg	traceId	rpcId

结构化日志格式

时间戳	异常类	异常方法	异常行号	错误码	错误信息	请求id	堆栈
timestamp	clazz	method	lineNumber	errorCode	errorMsg	traceId	exceptionStack

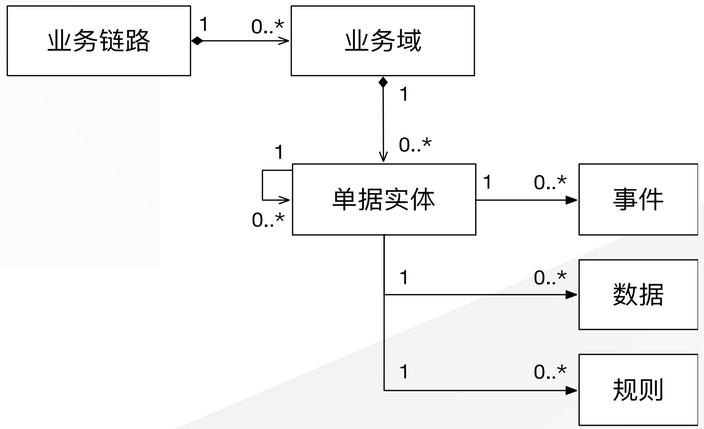
2）以工单作为突破口，建设『单笔全链路排查』能力



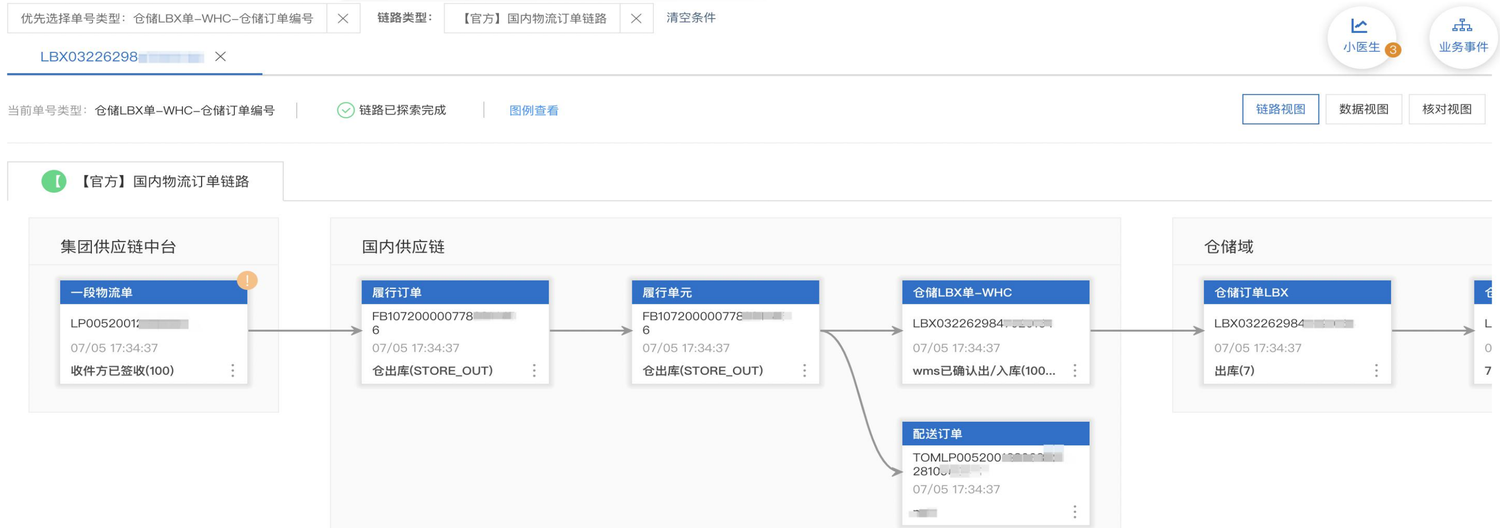
菜鸟的工单排查痛点分析

1. 每笔工单都是一次全链路排查需求
2. 业务团队自身的排查能力，只能查本域的数据，无权限跨域查询
3. 菜鸟有N种单据模型，交易单、物流单、电子面单、仓储单、运单等，单一业务域无法串联全局
4. 数据存储异构，全链路各个节点的数据存储介质不同，需要横向平台打通所有异构数据源

菜鸟业务全链路排查平台建模



产品形态



3）『单笔核对监控』，验证上下游数据一致性

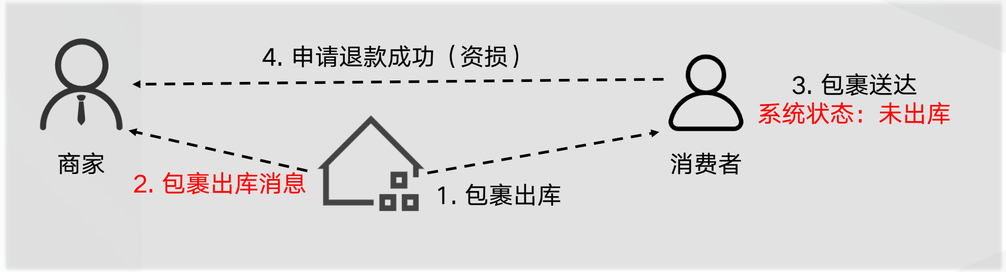
指标类监控：一段时间内对系统内指标的统计

- 用户层（访问速度、crash率、满意度等）
- 业务层（核心业务指标监控，如订单量）
- 应用层（JVM相关、中间件等）
- 系统层（CPU、load、mem、network等）
- 基础设施层（机房、网络、服务器等）

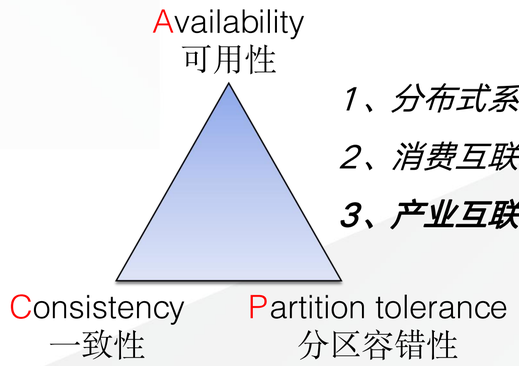
监控系统的黄金指标（Google SRE）：

- 流量，qps
- 延迟，rt
- 成功率，99.99%
- 饱和度，性能拐点

99.99%成功率 * 每天1亿包裹 = 每天1万包裹异常



真实故障案例，需要通过一致性核对监控发现异常

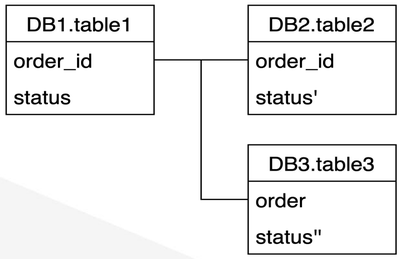


1、分布式系统最多满足CAP两项

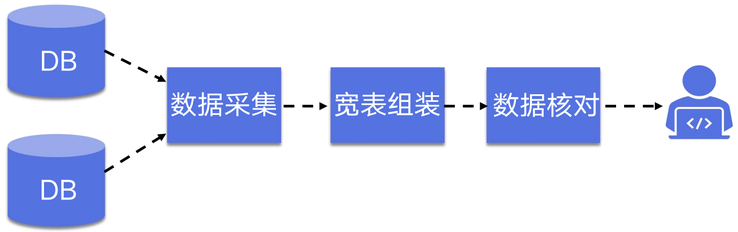
2、消费互联网侧重于AP，系统可用性：指标监控

3、产业互联网侧重于CP，数据一致性：核对监控

分布式系统数据模型



数据链路图

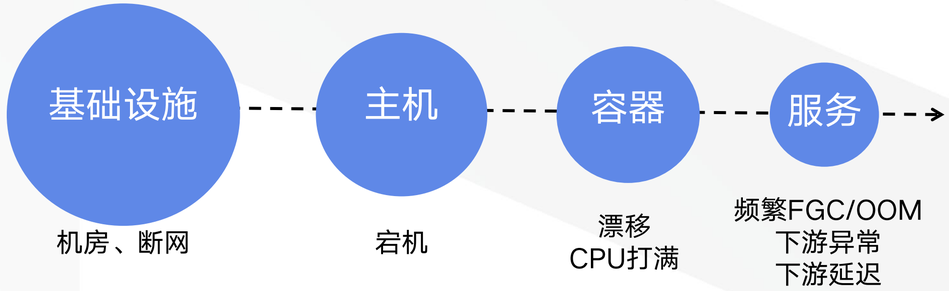


日均核对量：数百亿条数据

4）『单笔异常注入』能力，反向验证高可用效果

常见的异常注入能力（爆炸半径从大到小）

ChaosBlade, ChaosMonkey, ChaosMesh, ...



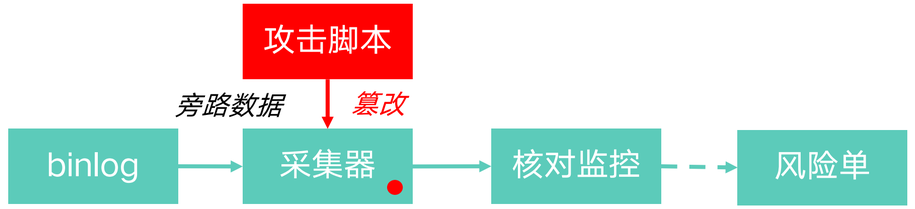
混沌工程在产业互联网落地的挑战

1. **异常注入不够真实**，现有的攻击能力目的是让“系统不可用”，产业互联网更关注有没有“异常的单子”（数据一致性）
2. **真实异常不敢注入**，用户对服务失败的容忍度更低。比如用户寄件/取件场景，每次异常注入都可能造成用户体验折损，甚至是工单或投诉。（要求无损）

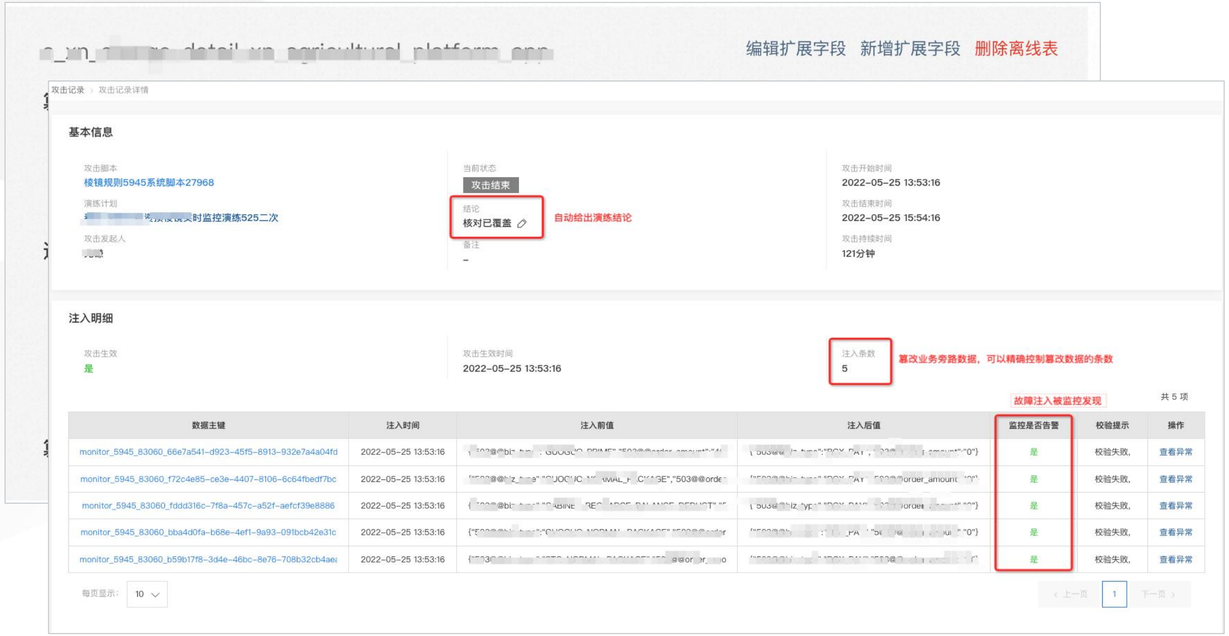
篡改旁路数据
验证监控有效性

构造测试流量
精准控制爆炸半径

单笔数据篡改核心模型



如果没有告警则生成风险单，监控缺失或失效



目录

产业互联网更需要『鲁棒性架构』

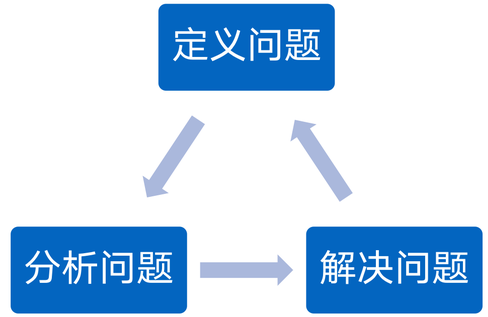
鲁棒性Robustness

系统在执行过程中，面对错误、异常及非法输入，仍能正常运行的能力 – *维基百科*

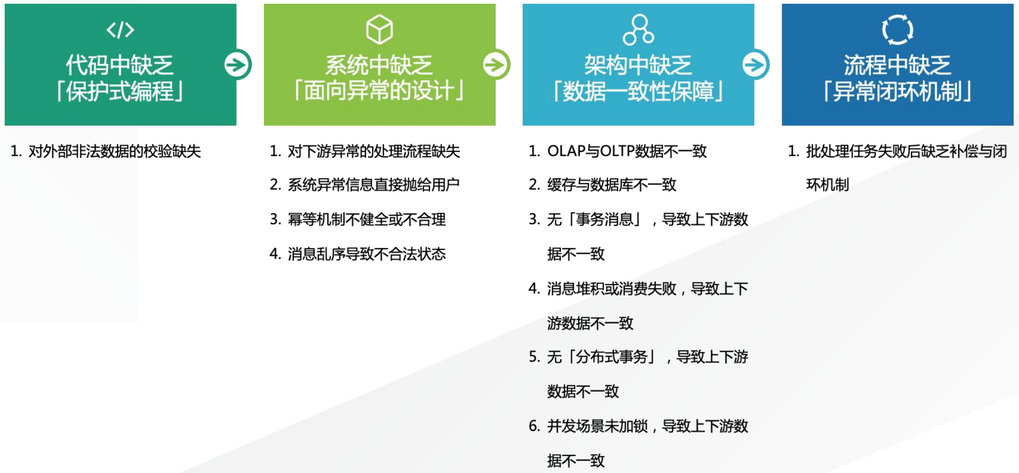
菜鸟鲁棒性架构升级的挑战

- 菜鸟很多因鲁棒性不足导致的工单，与bug类持平
- 重推、订正类临时解决方案非常简单
- 沉底解决需要投入成本做架构升级
- 系统鲁棒性比较抽象，一线研发对此无统一认知

整体解决思路

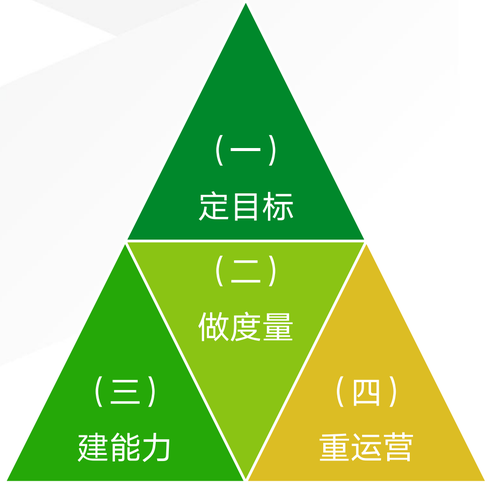


分析问题：从2k+工单中抽象出四大类12小类



用“剥洋葱”方式确保抽象的全面性，通过架构师、SRE、一线研发同学评审校准

解决问题：鲁棒性架构落地的四大策略



一、挑战型目标牵引项目，鲁棒性工单一年减半

二、建立度量体系，以特殊关键词作为观测指标

三、技术工具建设，让鲁棒性架构更简单

1. 建设针对下游抖动类异常的重试组件
2. 建设幂等、分布式锁等通用组件
3. 治理异常日志，提前发现异常，避免工单流入
4. ...

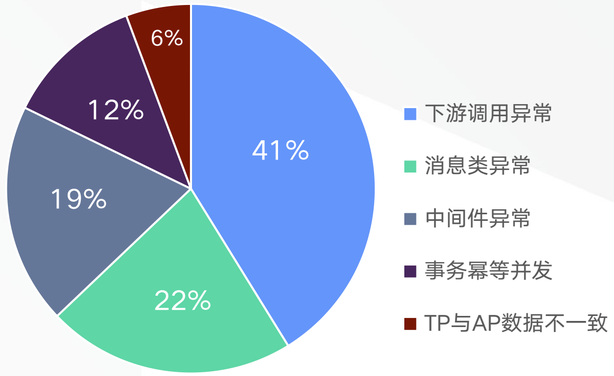
四、重视技术运营，鲁棒性小课堂、每日工单跟进

加深了一线研发对鲁棒性架构的认知，鲁棒性工单数量已有收敛趋势

『防抖组件』，让重试更简单

鲁棒性工单细化分类占比分析

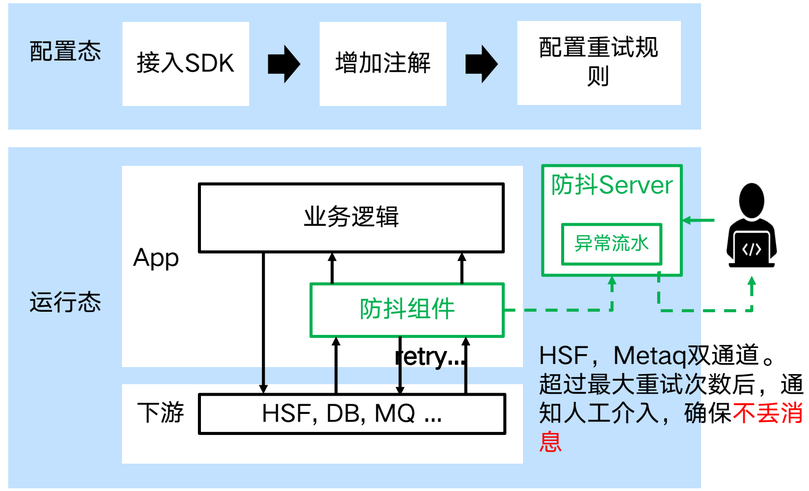
鲁棒性工单中，“重试”关键词占比24%



功能的丰富性和产品易用性是通用问题

重试方式	优点	缺点
硬编码	• 实现快	• 大量重复代码
MQ 自发自收	• 方案成熟	• 引入额外中间件 • 最大重试次数限制
任务调度中间件	• 强大的重试能力	• 方案过重 • 非异步场景，仅失败重试
Retry框架	• 复用性强	• 产品化能力弱 • 一般仅支持本地重试

菜鸟防抖组件解决方案



方便的本地接入流程

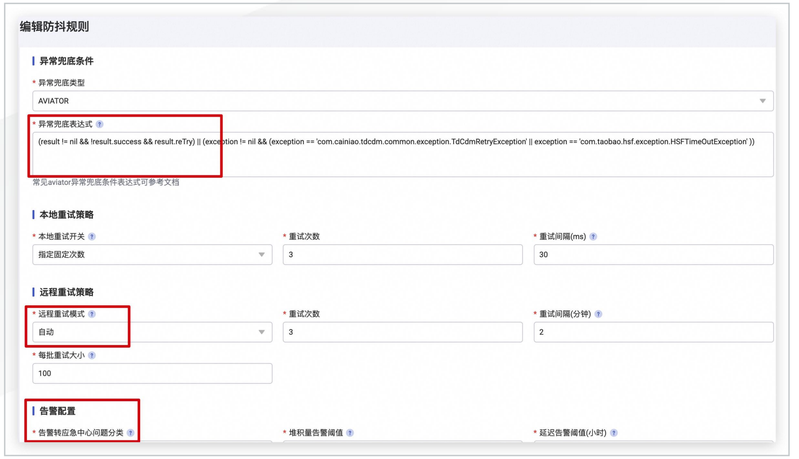
```
@Component
public class SpringBeanImpl implements ISpringInterface {

    @Resource
    private HelloService helloService;

    //增加Retry注解即可，value必填表示防抖规则编码，要求当前应用内唯一
    @Retry(value = "SpringBeanImpl.returnSuccess")
    @Override
    public ResultDTO<UserDTO> returnSuccess(UserDTO u1) {
        u1.setUsername("Hello, " + u1.getUsername());
        return ResultDTO.create(u1);
    }

}
```

丰富的云端动态配置能力

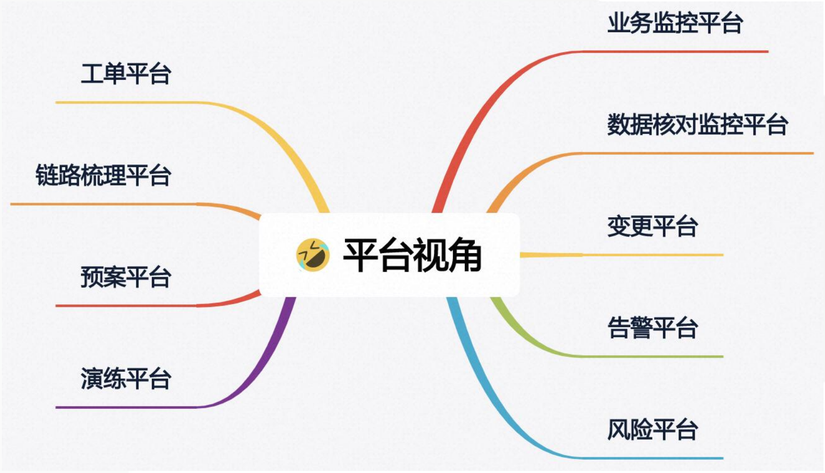


让重试更简单，累计避免近百万次异常卡单

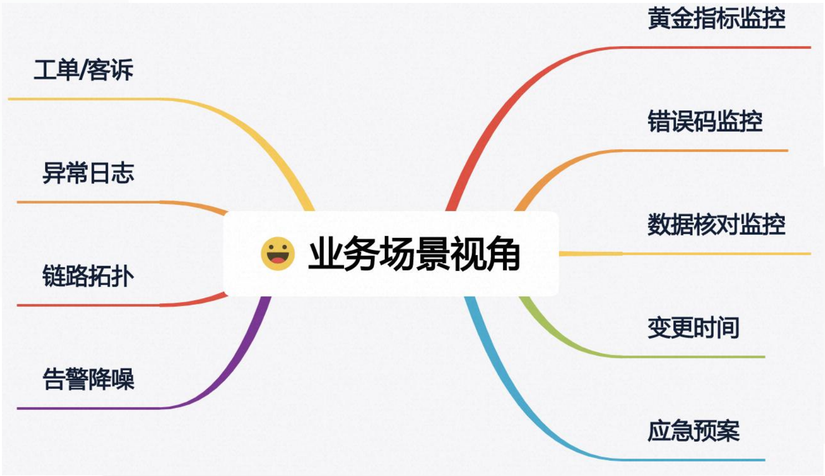
目录

1）以『业务场景』为视角的观测产品

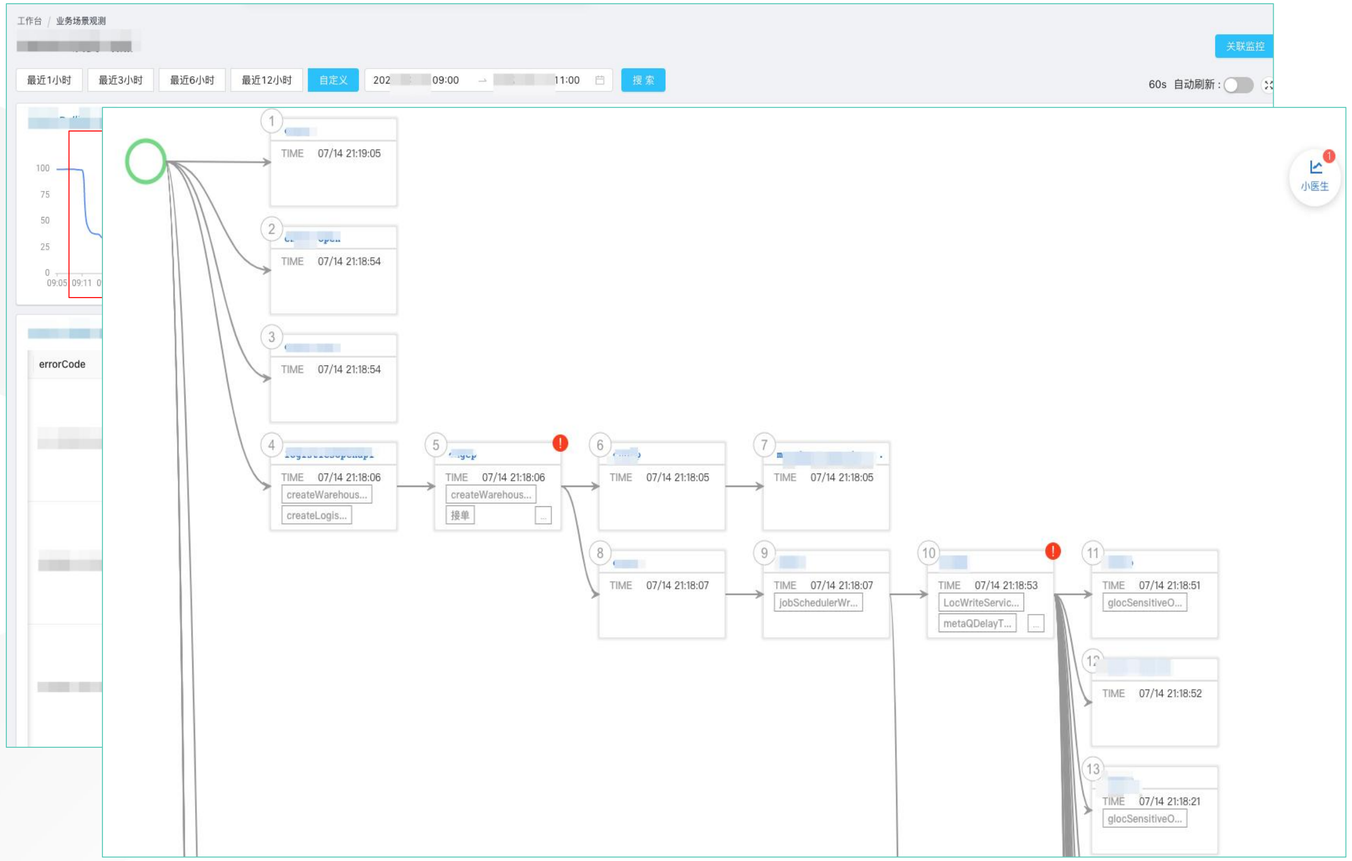
工具能力：10+高可用平台，要啥有啥



用户任务：故障定位时如何快速汇聚数据？



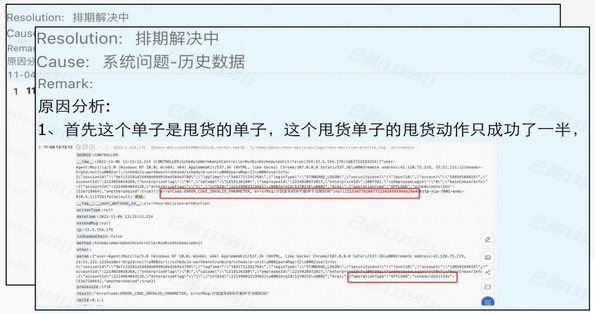
产品理念：减少用户完成「任务」的步骤，对已有平台做解构与重构



用历史故障的数据做推演，新人也能在3分钟内定位到原因

2）面向『治理流程』的异常日志治理产品

“冰山”之下的异常日志风险

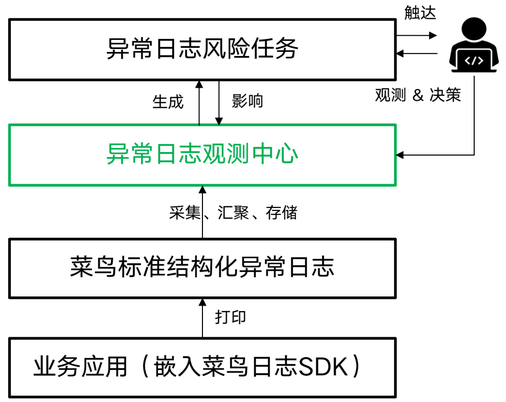


线上应用每天Exception“数以万计”，治理难度大

福格行为模型：行为=动机+能力+行动提示

- **动机✓**：对工单、故障的“恐惧”，希望服务更稳定
- **行动提示×**：异常日志风险主动触达研发同学
- **现有产品能力×**：弱，仅有观测能力
  - 异常噪声大，大量无需治理的异常，决策难，降噪成本高
  - 异常要分类处理，系统类异常需要消灭，业务类异常要观测
  - 治理噪声成本高，改代码、测试、发布、灰度至少1天

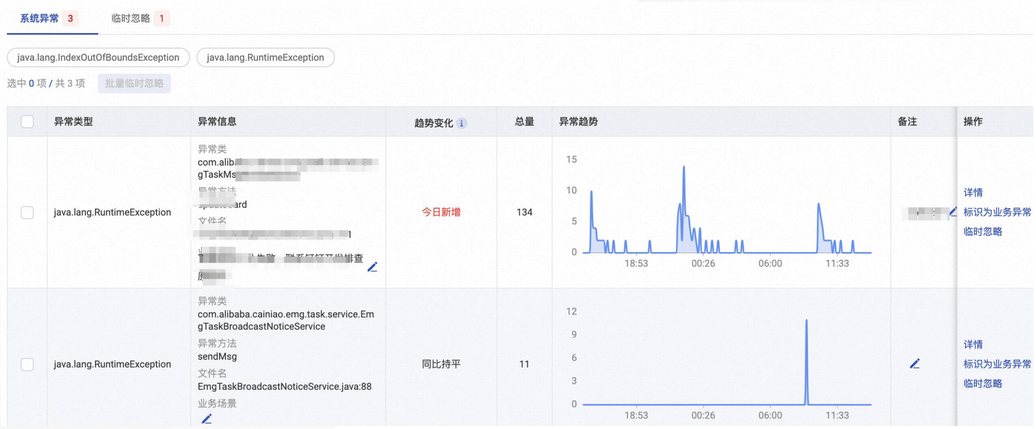
面向「治理」流程的核心模型



低成本是产品的成败关键

- **低成本接入：**
  - 不影响已有日志打印
  - 不需要修改已有监控
- **低成本使用：**
  - 一键治理，不发代码
  - 多维度数据辅助决策
  - 集成到变更风控流程中
- **低成本支出：**
  - 低于￥10/月（采样存储）

产品形态



整体回顾与总结

回顾

1.『产业互联网』：用互联网的技术做好产业

2.『单笔高可用』：单笔订单或单笔请求都要可用

3. 单笔高可用『工具能力』：日志、排查、监控、验证

4. 产业互联网的『鲁棒性架构』：面向失败的架构设计

5. 菜鸟高可用『产品建设』实践：减少用户任务步骤

总结

1. 所有的理念和工具都有其适用场景，不用迷信

2. 对自身业务场景的理解和抽象，是做好高可用的前提

3. 高可用是复杂问题，解决复杂问题，需要体系化拆解

4. 尽可能在上游（架构）解决问题，成本更低，效果更好

微信关注“菜鸟技术星球”

获取文稿版，还有更多高质量技术文章

QCon 全球软件开发大会 CAINIAO 菜鸟

微信关注“菜鸟技术星球”

获取文稿版，还有更多高质量技术文章